International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Weapon Detection Using Deep Learning: HLD and LLD Documentation

Preethi V M

Abstract

This paper presents a comprehensive study and implementation of a deep learning–based weapon detection system using state-of-the-art convolutional neural network (CNN) architectures and object detection algorithms such as SSD-MobileNet, Faster R-CNN, and YOLO. The system aims to identify and locate weapons in real-time video streams with high accuracy and speed. The methodology includes dataset preparation, model training, architecture design, and performance evaluation based on metrics such as precision, mean average precision (MAP), and F1 score.

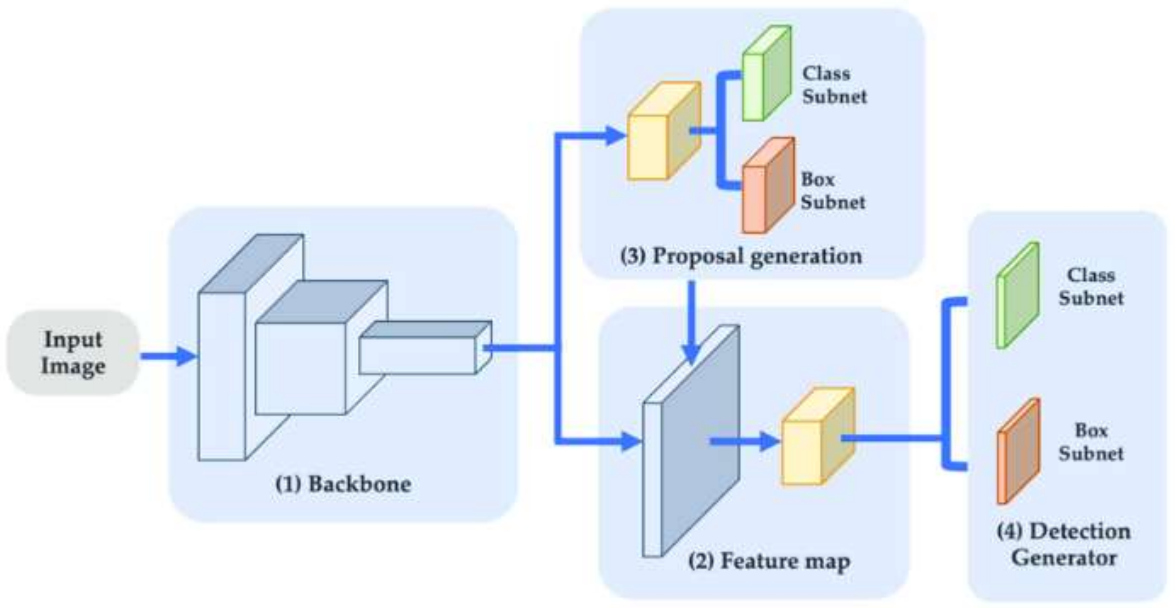
Keywords

Deep Learning, CNN, YOLO, Object Detection, Weapon Detection, Computer Vision

1. Introduction

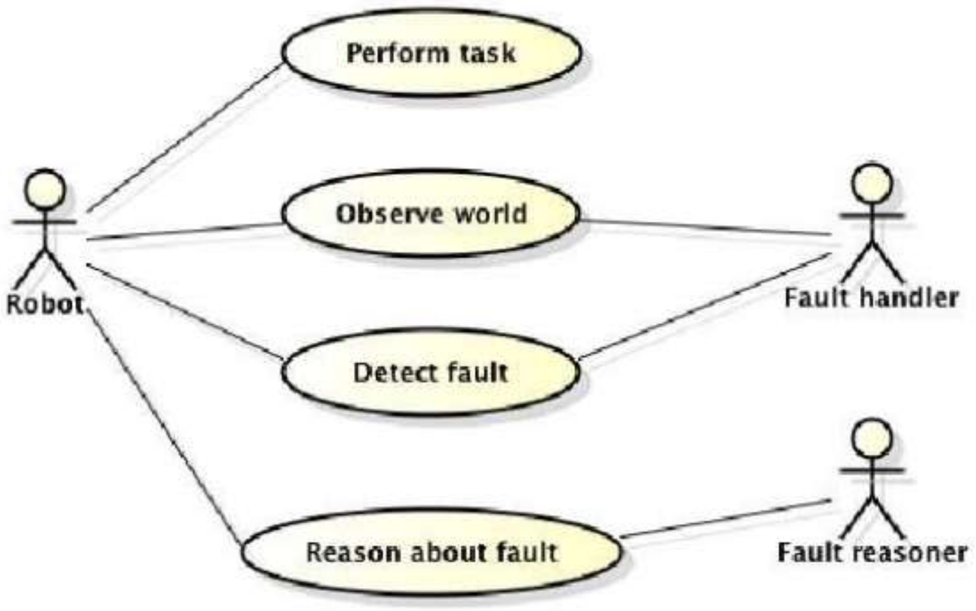
Weapon detection has become an essential component in public surveillance and modern security infrastructure. Deep learning techniques, particularly CNN-based architectures, have significantly improved object detection accuracy and performance. This research implements multiple classification and detection models and evaluates their efficiency under various scenarios.

2. System Architecture:



Deep learning is a branch of machine learning inspired by the functionality andstructure of the human brain also called an artificial neural network. The methodology adopted in this work features the state of art deep learning, especially the convolutional neural networks due to their exceptional performance in this field. Theaforementioned techniques are used for both the classification as well as localizingthe specific object in a frame so both the object classification and detection algorithms were used and because our object is small with other object in background so after window/classification and region proposal/object detection algorithms were used, and these techniques will be discussed later in this section. We had started by doing the classification using different deep learning models and achieved good precision but for the real-time scenarios, the low frame per seconds of classification models were the real issue in implementation. Oxford VGG, Google Inceptionv3 and InceptionResnetv2 were trained using the aforementioned approach. To achieve high precision, increase number of frame per seconds and improve localization, we moved to the object detection and region proposal methods. The different state of the art deep learning models for object detection were used and the results were compared in terms of precision, speed, and standard metric of F1 score. State of the art deep learning based SSDMobileNetv1, YOLO, FasterRCNN-InceptionResnetv2, and YOLO were trained and tested. Different datasets were made keeping in mind theclassification and detection problem as both have a separate requirement for performing the tasks to achieve high accuracy, mean average precision as well as frame per second for the realtime implementation. To understand object classification and detection let us first briefly understand object recognition as both the aforementioned types come under the umbrella of this and combined classification and localization make detection possible for any kind of detection problem giving class name as well as the region where our desired object is in the frame.

3. Use Case Diagram



A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent: Scenarios in which your system or application interacts with people, organizations, or external systems Goals that your system or application helps those entities (known as actors) achieve and The scope of your system

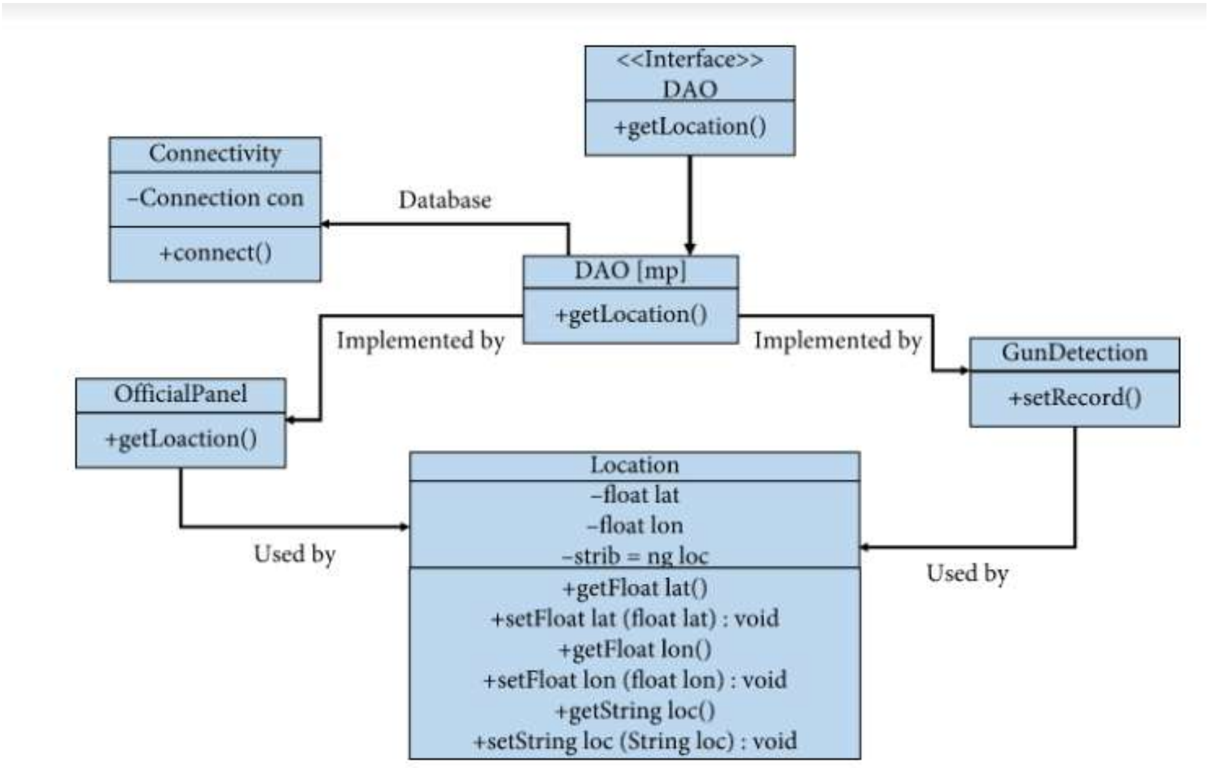
4. Class Diagram

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A classwith three sections, in the diagram, classes is represented with boxes which contain three parts:

The upper part holds the name of the class

The middle part contains the attributes of the class

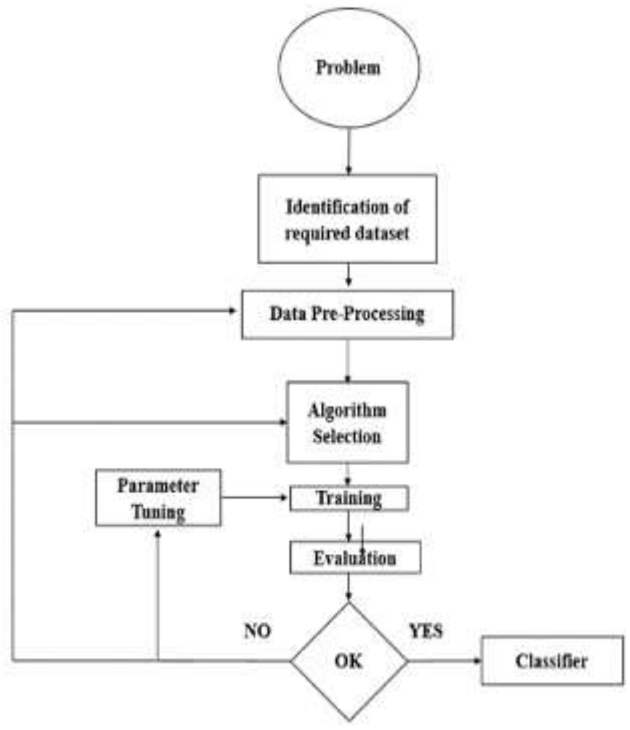
The bottom part gives the methods or operations the class can take or undertake.



5. Activity Diagram

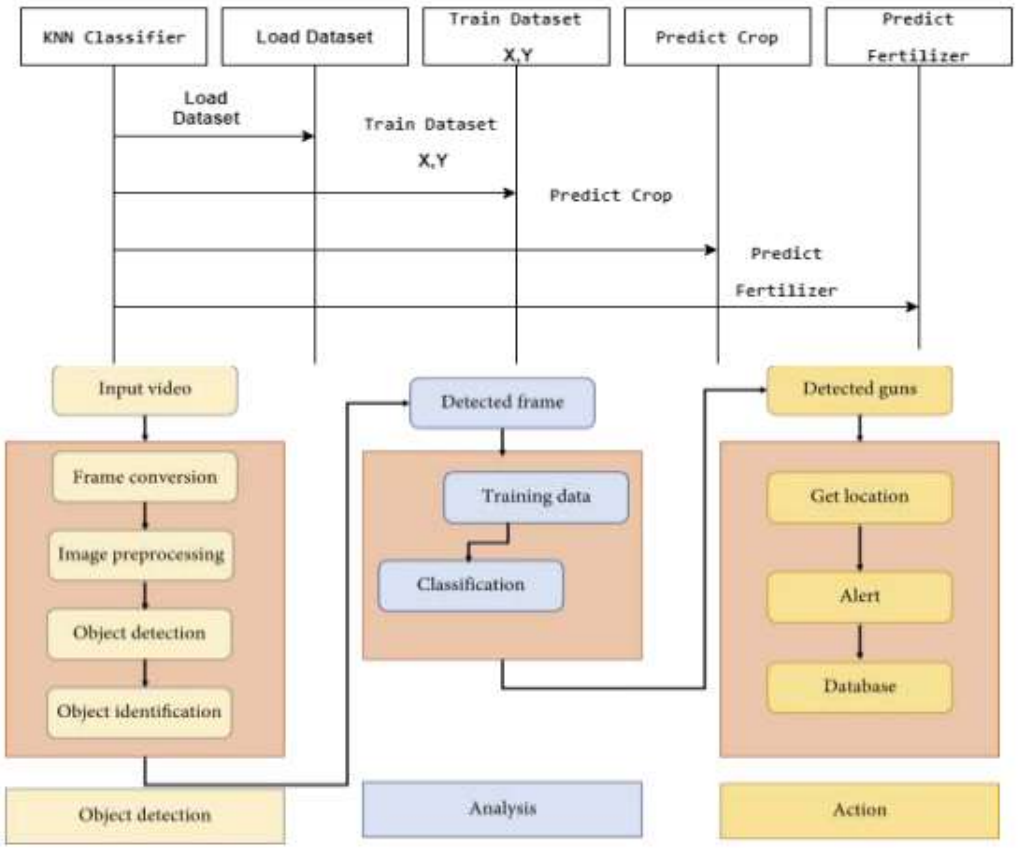
Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something.

The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.



6. Sequence Diagram

one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically realizations in the Logical View of the system underA sequence diagram is a kind of interaction diagram that shows how processes operate with development. Sequence diagrams are sometimes called event diagrams, eventscenarios, and timing diagrams.



7. Methodology

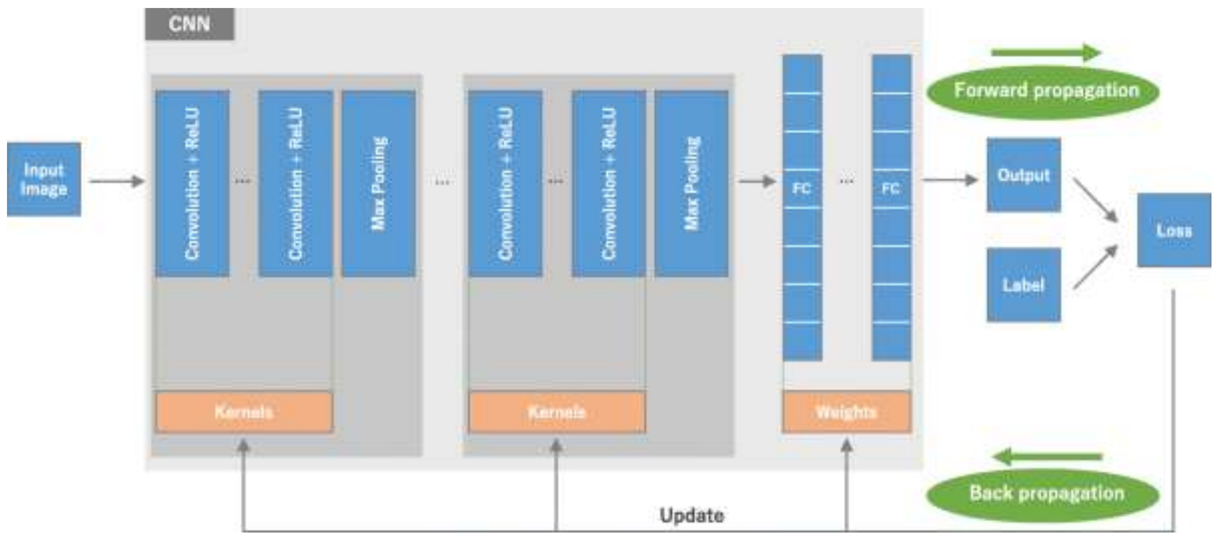
Deep learning techniques were used for both object classification and detection. Initial classification models such as VGG, InceptionV3, and InceptionResNetV2 were trained, but real-time performance limitations led to the adoption of object detection approaches. Models such as SSD-MobileNet, Faster R-CNN, and YOLO were evaluated and compared.

Algorithm:

7.1 Convolutional Neural Network (CNN)

Convolutional neural networks (CNN) – the concept behind recent breakthroughs and developments in deep learning.

Computer vision is a very popular field in data science, and CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short-term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular. These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition, etc. They have hence been widely used in artificial intelligence modeling, especially to create image classifiers. This article will introduce you to the concept of image classification using CNN and show you how they work on various datasets.



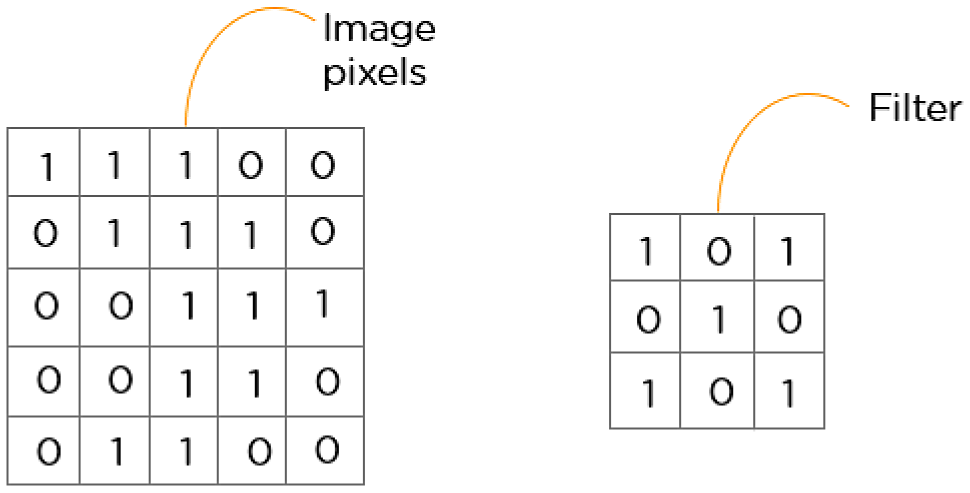
A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

1. Convolution layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer

Convolution Layer

This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.

Consider the following 5x5 image whose pixel values are either 0 or 1. There's also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.



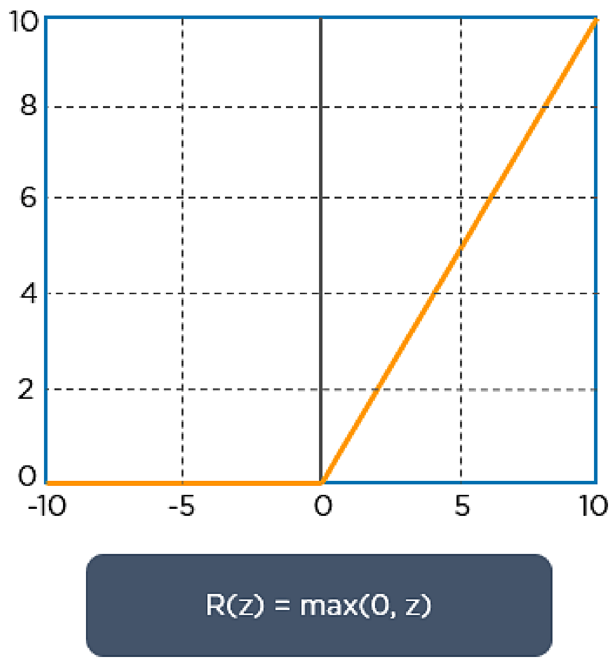
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

ReLU layer

ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Below is the graph of a ReLU function:



The original image is scanned with multiple convolutions and ReLU layers for locating the features.

Pooling Layer

Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.

The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.

The pixels from the image are fed to the convolutional layer that performs the convolution operation

It results in a convolved map

The convolved map is applied to a ReLU function to generate a rectified feature map

The image is processed with multiple convolutions and ReLU layers for locating the features

Different pooling layers with various filters are used to identify specific parts of the image

The pooled feature map is flattened and fed to a fully connected layer to get the final output.

7.2 YOLO Algorithm

YOLO (You Only Look Once) is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015. It is a single-stage object detector that uses a convolutional neural network (CNN) to predict the bounding boxes and class probabilities of objects in input images. YOLO was first implemented using the Darkent framework.

The YOLO algorithm divides the input image into a grid of cells, and for each cell, it predicts the probability of the presence of an object and the bounding box coordinates of the object. It also predicts the class of the object. Unlike two-stage object detectors such as R-CNN and its variants, YOLO processes the entire image in one pass, making it faster and more efficient.

YOLO has been developed in several versions, such as YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, and YOLOv7. Each version has been built on top of the previous version with enhanced features such as improved accuracy, faster processing, and better handling of small objects.YOLO is widely used in various applications such as self-driving cars and surveillance systems. It is also widely used for real-time object detection tasks like in real-time video analytics and real-time video surveillance.

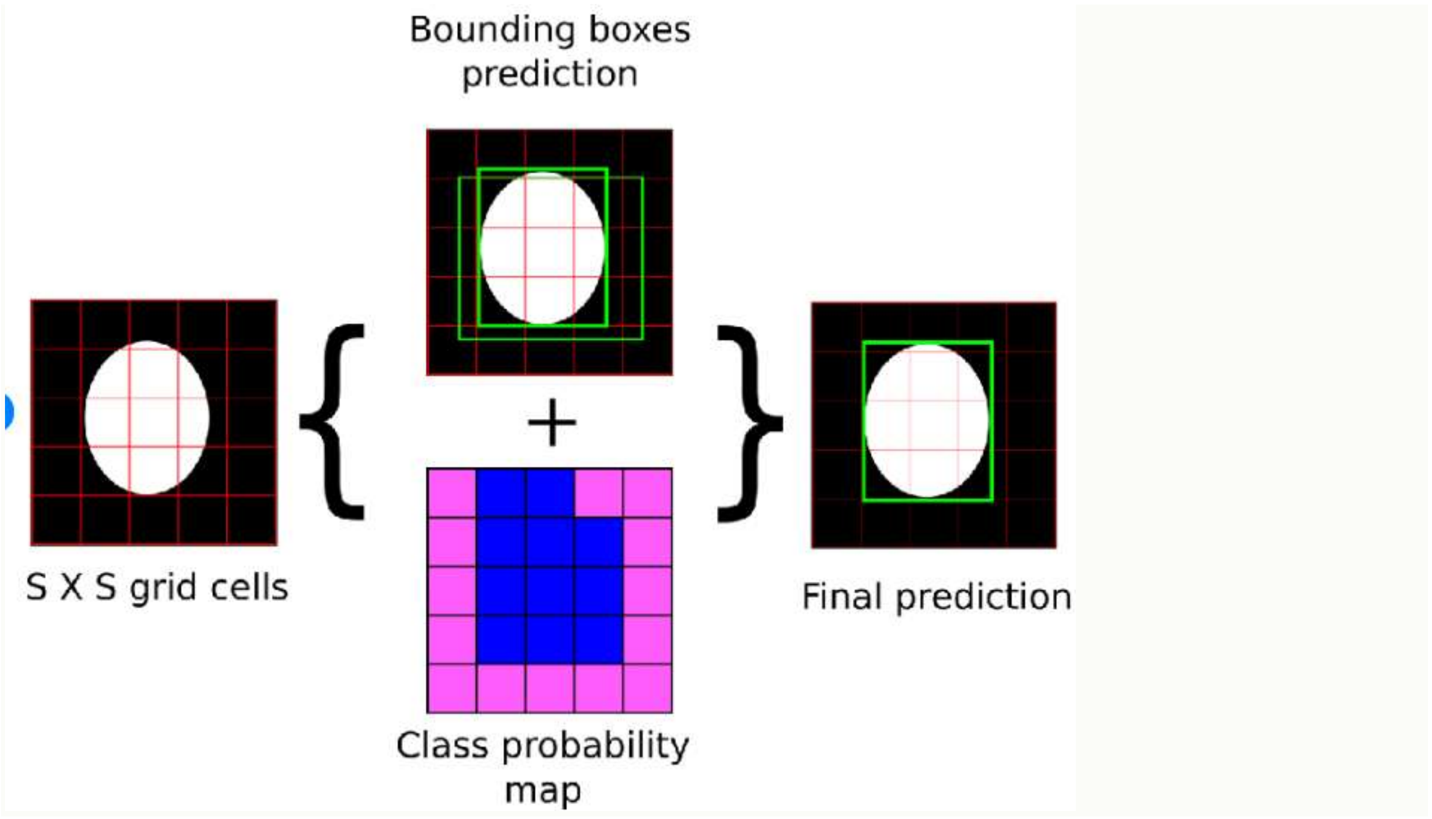
The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object. The process of YOLO can be broken down into several steps:

1. 1. Input image is passed through a CNN to extract features from the image.
2. 2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.
3. 3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.
4. 4. The output of the network is a set of bounding boxes and class probabilities for each cell.
5. 5. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.
6. 6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

YOLO v5 was introduced in 2020 with a key difference from the previous versions, which is the use of a more efficient neural network architecture called EfficientDet, which is based on the EfficientNet architecture. EfficientDet is a family of image classification models that have achieved state-of-the-art performance on a number of benchmark datasets. The EfficientDet architecture is designed to be efficient in terms of computation and memory usage while also achieving high accuracy.

Another important difference is the use of anchor-free detection, which eliminates the need for anchor boxes used in previous versions of YOLO. Instead of anchor boxes, YOLO v5 uses a single convolutional layer to predict the bounding box coordinates directly, which allows the model to be more flexible and adaptable to different object shapes and sizes.

YOLO v5 also uses a technique called "Cross mini-batch normalization" (CmBN) to improve the model's accuracy. CmBN is a variant of the standard batch normalization technique that is used to normalize the activations of the neural network.



8. Results and Discussion

The models were tested using custom datasets tailored for weapon detection. YOLO-based models achieved the highest frame-per-second performance, making them suitable for real-time surveillance systems.

9. Conclusion

This study concludes that YOLO-based architectures offer superior real-time performance while maintaining high accuracy in detecting weapons. Future enhancements include expanding datasets and improving detection precision for small or partially occluded objects.

References

1. Joseph Redmon et al., YOLO: Real-Time Object Detection.
2. Ren et al., Faster R-CNN.
3. Szegedy et al., Inception Architecture.
4. Howard et al., MobileNet Architecture.